A SMARTPHONE-BASED ENERGY SAVING NAVIGATION SYSTEM

BY

CHENJI PAN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Professor Tarek Abdelzaher

ABSTRACT

This thesis describes a smartphone-based vehicular GPS navigator by using an *energy saving* location sensing mode and it reduces navigation energy needs by more than an order of magnitude. The thesis will go over the theory and mode behind it and focus on the implementation details and the survey and user study we did during the research. Besides, the extensive work on combination between eNav and GreenGPS [1] will also be covered in the thesis. Navigation applications becomes the killer application with the popolarization of smartphone. However, navigation could be one of most energy consumption tasks in our smartphone since smartphone based GPS navigation applications available in the markets set the highest GPS sampling while working to guarantee the best quality of navigation. Nowadays, USB charging interface is available in most of cars and the drivers usually will keep the phone plugged-in during navigation to keep the phone have enough battery for other usages. But a charging cable may not be always available during trips, especially in hire cars or cars borrowed from friends. And according to a survey we conducted based on 500 respondents, more than 90% would like to have a energy-saving navigation application. So here we propose our eNav, a novel smartphone navigation application, which exploits phones' local low-energy sensors like accelerometer and gyroscope to estimate the approximate location and only samples GPS data when close to the way-points in the route. Our eNav system has been implemented from scratch on Android and 33 drivers have tried it in our user study. Our evaluation shows that this approach increases battery life during navigation by more than 80%.

To my parents, for their love and support.

ACKNOWLEDGMENTS

First, I would like to thank my advisor, Tarek Abdelzaher, for his patience, support, advices and his trust. He showed me how to be a great researcher by his passion in his work.

Also the work would not be possible without Shaohan Hu. He made the most contribution in the theory part of this work. It is really nice to work with him. We have worked together for this work until late nights for several months. The encouragement from him inspires me to finish the work.

Last but not least, I would like to thank all the partners in this work. Their suggestions and ideas make the work better.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

This thesis introduces our energy-saving GPS navigation system, eNav. In our survey, about 70% of people use phone-based GPS navigation application when driving, which shows that more and more people are using their smartphone for navigation during driving. This kind of trend can also be reflected in decline in GPS navigation devices sales [2] and navigation application like Google Maps are on the top 10 usage of all smartphone applications [3].

GPS module is one of the most power consuming part on our phones [4] since it has to communicate with three or four satellites and it prevents the smartphones from entering into sleep state. By running Google Maps to navigation, a fully charged smartphone can be exhausted in a few hours. On the other hand, navigation applications is one of the most useful and popular utilities on our phone. With the popularization of smartphones, drivers are more likely to use their phones as navigator rather than buy a functional singleness navigation device. To avoid consuming too much energy before arriving the destination, drivers will keep the phone plugged-in during the trips. However, drivers could forget to take the charger or charging line with them, especially when traveling in a hire car in a new city. Bringing the charger all the time is not quite convienient as the phone and wallet has filled our pockets. A better solution or say as a backup plan, a power-saving navigation application is well-needed choice while navigating.

Navigation service is more in need when travelling in a unfamilar city rather than in local town in daily life. People are likely to hire a car from car rental services such as Hertz, Enterprise, Avis and Budget after they arrive the airports or railway stations while travelling. According to [5], about 1.8 million cars are available in the US alone for rent, serving more than one hundred million trips annually. Since bringing a charger during trip is not convienient and easy to forget, our energy-saving system eNav could serve as a good choice while driving on way to some places of interest. And for better understanding the demand for such a kind of system, my partners and

I have conducted a survey on 500 respondents on CrowdFlower.com [6] between Nov 20th 2013 and Jan 5th 2014, which will be detailed describe in later chapter.

One of my partner is a experienced driver and travel around the world a lot and he is also a heavy smartphone user. He only use the phone to navigation when he is travelling in a new city and is annoyed by the short lifetime of the phone. So we digged into the research in the navigation system and found that the navigation system like Google Maps trend to return the route with limited number of critical way-points, which is usually the turning point at crossroad between two roads. Moreover, I found that my father, a man with over 20 years driving experiences and drived in old days when there were no navigator devices or it was not popular as today, he used to and still keep the habits to check the route on the paper map and only remember the key turing points in his mind before leaving home. Our research on navigation system and this kind of old-fashion navigation method inspire my partners and me to think about an energy-efficient way for navigation and a question came into our mind: Do we really need to know where we are or say sample the GPS geo points during the whole trip? For example, if our destination is only on the other end of the road where the car is and driving 2 miles straight forward along the road, most of us do not need the navigation system under this circumstance. So how about we divide the complicated route into segments and each segment represent a straight forward route from one end to another end. After the division, we then assemble them by a short period of GPS samplings. In other words, when the car is close to the way-point, missing a way-point could result in a great cost. Our eNav system will sampling the GPS data during these critical period as the old-fashion driver slow down to check the road sign to make sure not to miss the turning.

With the question and idea above, we develop our energy-saving navigation system eNav. However, while driving on one segment, the driver have to be noticed when the next way-point is close. So eNav have to estimate the approximate location without GPS samplings. Techniques like dead-reckoning [7] could be used here to solve the problem. Dead-reckoning computing the locations by intergrating the motion measurements during a peroid and low-energy consuming sensors like accelerometer and gyroscope provide a way to get the motion measurements with a much lower energy cost. Nevertheless, the sensors on the smartphone is not that accurate and high-quality as in traditional dead-reckoning system. Besides, the phone's orientation to the car's driving direction is unknown and changes in each drive.

We build our energy-saving location sensing mode to solve the challenges and problems described above and develop our eNav navigation system. To check the feasibility and usefulness of our system, a study which includes 33 drivers has been conducted as our empirically evaluation. To overcome the noise caused by known route, the participants are driving to some destinations they were not familiar with in the experiments. The results of the user study shows that the eNav system saves over 80% energy energy and no driver has ever missed any navigation way-points.

This thesis will cover both the theory and implementation of eNav system. Since the theory has been detailedly described in my partners' and my collabrative paper[8], this thesis will only go over the theory generally and focus on the implementation details. Moreover, the extension work about combination between GreenGPS [1] will be covered. GreenGPS is an efficient-fuel-consumption route service based on participatory sensing data. Rather than providing the shortest route like Google Maps did, GreenGPS returns the most fuel-efficient route from the start to the destination.

The rest of the thesis is organized as follows. Chapter 2 illustrates the survey and results we conducted on CrowdFlower.com. After that, Chapter 3 will go over the theory and mode generally. Chapter 4 describes the detailed working flow of our eNav system. Chapter 5 will cover the implementation details and codes. Chapter 6 gives the evalution of our system and the results of our user study. Chapter 7 discusses about the combination work between GreenGPS and our eNav system. Chapter 8 we discuss related work, and finally, Chapter 9 concludes the thesis. A survey questions sheet on the need of system like eNav is posted in Appendix.

CHAPTER 2

MOTIVATION AND DEMAND STUDY

In this Chapter, we will discuss the survey we conducted on CrowdFlower.com and the motivation based on the results of the survey.

2.1 Survey

The idea of eNav came from my partners' and my past driving experience. But to make it as a real working system and public, we have to figure out if the phone based navigation system is popular and common in our daily life and whether they would like to use this kind of system to save the battery of their phone and if they think it would be a useful system for them. Besides, to save more energy, we also add an extra-energy-saving scheme for eNav system, which is making the eNav work under screen off status. Therefore, the navigation instructions are cued by voice messages rather than visual. To understand the needs for the scheme, questions like how they think about navigation based on voice instructions will be covered in our survey.

To figure out the people's thoughts on these questions, my partners and I made a survey with 13 questions and posted it on CrowdFlower.com during Nov 20th 2013 and Jan 5th 2014. We ended the survey after receivng 500 valid responses. Except the demographics questions, all questions are multi-choiced. CrowdFlower.com [6] provides service for conducting survey and anyone who register with credit card could answer questions to make benefits. However, in order to focus our targets on people who drives car and to keep the responses to be more reliable and with less noise, we set five randomly-placed repeated questions with reordered choices. The questions are all showed in Appendix. To increase the incentive of the attending, $0.5 dollars are given to each participant who finish the survey and CrowdFlower.com automatically prevent same person to repeatedly answer the same sheet.

In the 500 responses, the demographics are showed as follows:

1. Gender: 38.6% of the respondents are male and 61.4% are female.
2. Age: ranging from 18 to 64 years old. The detailed distribution histogram is showed in Fig. 2.1.
3. Origin: vary from 385 cities in 48 states in United States.
4. Driving Frequency: 68.6% of them drive every day, 21.8% drive once or twice a week and the remaining 9.6% drive rarely.
5. Driving for: 84.4% was for shopping, 69.6% was for occasional entertainment, 65.2% was driving to work and 46% was for long distance travel
6. How long: More than 75% would drive at least 15 minutes.

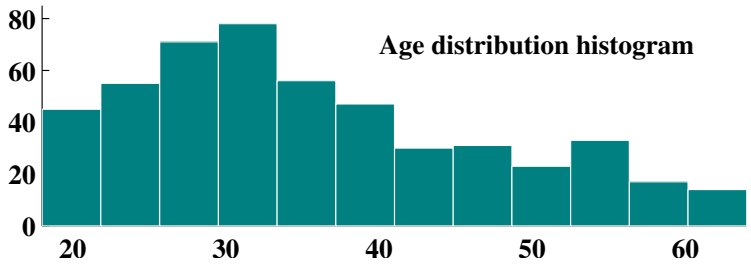


Figure 2.1: Age Histogram

On the problem about how common and popular to use smartphone as navigation device in the car, 80.4% of all the 500 respondents claimed to use GPS nagivation in the car and 73% said they used GPS navigation application in the smartphone. For situation we talked about in above (in hire cars), over 59.4% will use smartphone-based GPS navigation in a rental car. 37.4% of the drivers had experienced low battery status while using phone-based navigation during trip.

When being asked about if they were interested in energy-saving navigation, which is the question: *"Imagine a phone-based GPS navigation system with an optional battery-saving mode that turns off the screen when it is not needed (e.g., when you are not moving or far from your next turn-point). Voice navigation will continue at all times. How useful would you find this battery saving feature?"* 91% of the respondents would like to have an application which could provide

energy-saving navigation service. 1/3 of these people would love to set this mode as default no matter the battery is full or in low volumn.

For the third question we mentioned above, which is about comparison between voice instructions and visual cues, 75% of the participants preferred voice instructions to visual one or thought both are OK. Furthermore, under the low-battery situation, this ratio raised up to 81.8%.

2.2 Motivation

To find the reason behind the phenomenon that the results gave, we also calculate the correlation between the participants' demographics and their preference. There was a strong correlation between preferring voice instructions and interesting in system like eNav. Most of the people who are interested in voice instructions and eNav are frequent drivers (drive every day), which means we found a strong coorelation between frequent driver and preference in voice instructions and eNav. Besides, using GPS service on smartphone and high frequency of using charger in the car also have strong coorelations with the attributes above. The reason behind it is quite straightforward. People drives everyday are suffering the battery exhausty more often. Experienced drivers are more relaxed when driving, which made them be able to drive without heavily relying on the exact location all the time. The strong coorelation between high frequency of using charger in the car and loving to have a system like eNav might be caused by the inconvenience of charging the phone when driving in the car or at least it is not a preferable way.

According to the feedback of the survey, we can summarize three important observations.

1. The smartphones are commonly used for navigation when driving.

2. An energy-saving navigation mode is welcome and in need even if there was enough power in the phone.

3. Most drivers can accept screen off and only rely on voice instructions to navigation for energy saving.

With the results and analysis behind the strong coorelations, we have strong enough motivation to make our eNav to come true.

CHAPTER 3

THEORY AND MODE

In this chapter, we will generally go over the theory behind eNav system. The details about the theory is published in Shaohan Hu's and other co-authors' (includes me) paper[8].

3.1 Key Problems in Design

Traditional GPS applications set the GPS sampling rate to highest(1Hz) to guarantee the best navigation quality. Smartphone's system like Android and IOS can not enter into sleep mode when the GPS module is on, which cause the battery running out quickly. Since our aim is to build a energy-saving GPS navigation system, the GPS usage has to be reduced. So here comes to the first key problem: when to turn on and turn off the GPS? Simply reducing the sampling rate will lead to miss way-points, which is observed in our experiments. For example, decreasing from 1Hz to 0.012Hz would cause about 83.1% missing turns, which will significantly affect the navigation quality. Therefore, we need a solution that implements both reducing GPS usage and guarantee the navigation quality (not missing turns).

As we mentioned above, the navigation services like Google Maps usually return route with a limited number of way-points. Fig. 3.1 is a route from Chicago Union Stations to Chicago Indigo Hotel and we can see that there is only two way-points. Fig. 3.2 shows a route from Thomas M. Siebel Center for Computer Science of University of Illinois, Urbana-Champaign to University of Illinois Urbana-Champaign Willard Airport, which is a quite common route that students and faculties in computer science department of UIUC usually follow. We can see that whether it is in big city like Chicago or urban area like Urbana-Champaign, the route usually only contains limited critical way-points. Actually, we have try a lot of routes on Google Maps for various areas, most of the routes only contains less than five way-points.

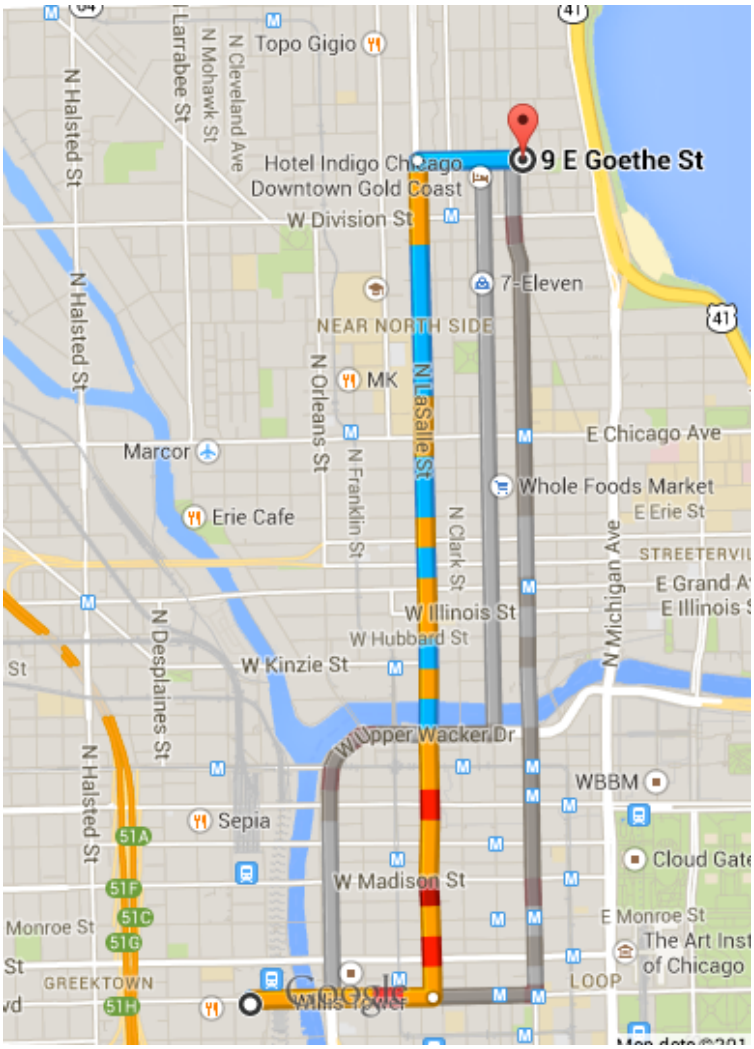


Figure 3.1: Route returned by Google Maps in Chicago

We've talked about one-segment route before and driver usually will not need GPS data all the way to the destination. Based on that idea, eNav system divide the whole route into several one-segment routes and only samples the GPS data at the joint (when the car is close to the way-point), which is would be like the red line part in the route showed in Fig. 3.3. While the car is still far away from next turning point, the GPS is turned off to save power. Here we have answered the first key problem.

However, since the GPS module is not activated during the driving on one of the segment, how could eNav know that the car is close to the way-point, which could be generalized to our second key problem: how to get approximate location of the car when the GPS is off?

Regarding to our second key problem, we recognize that the local sensors like accelerometer,

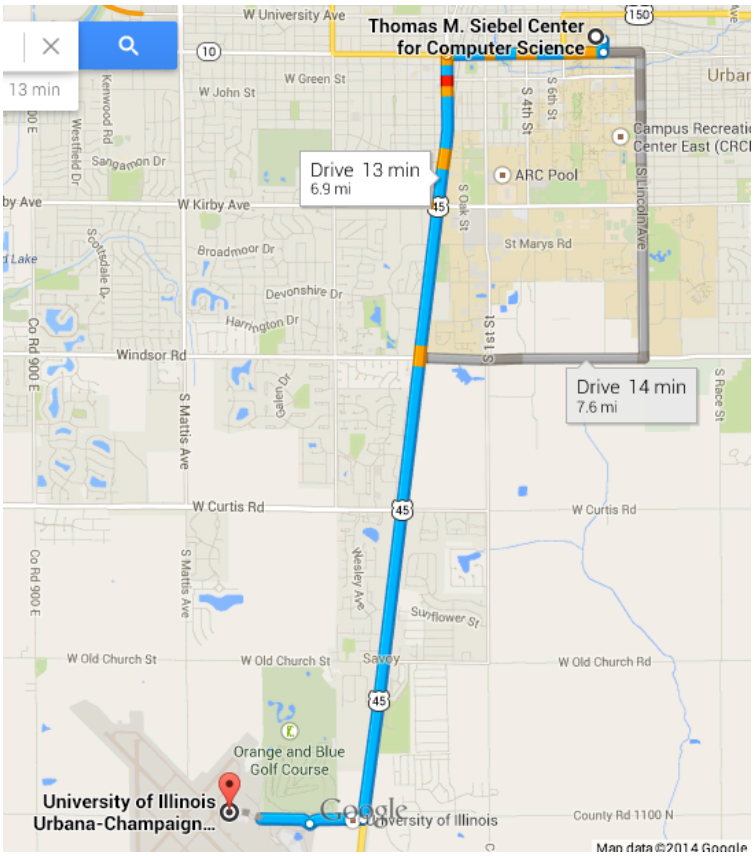


Figure 3.2: Route returned by Google Maps in Urbana-Champaign

gyroscope may help us get the inertial motion of the car. Since sensors like accelerometer, gyroscope only consumes a tiny amount of power(accelerometer's power is only about 0.00049W at 50Hz) compared to GPS module(0.15W at 1Hz). After getting the inertial motion of the car, techiniques like dead-reckoning [7] could be used to estimate the rough location of the car. Here we have answered two key problems that we met in implementating eNav. In next section 3.2, we'll talk about the feasibility of the answers we give here to the problems.

3.2 Feasibility Study

In this section, we'll focus on the feasibility study about our solution proposed above, which is mainly contributed by Shaohan Hu[8].

Before, we have discussed about using dead-reckoning by phone's local low power consuming sensors to estimate the car's location. Here we present the feasibility study, which means if the

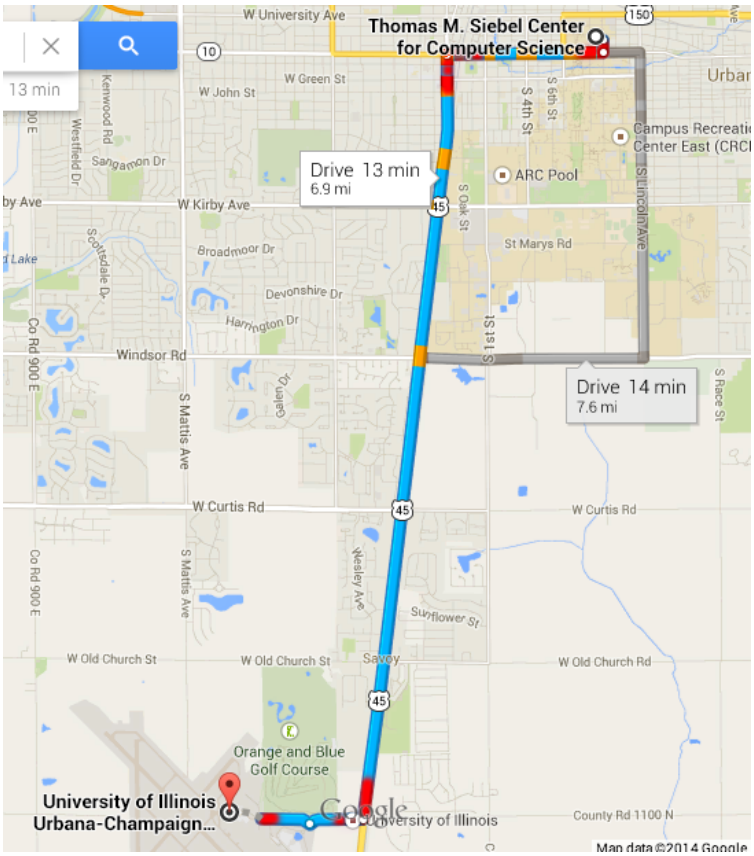


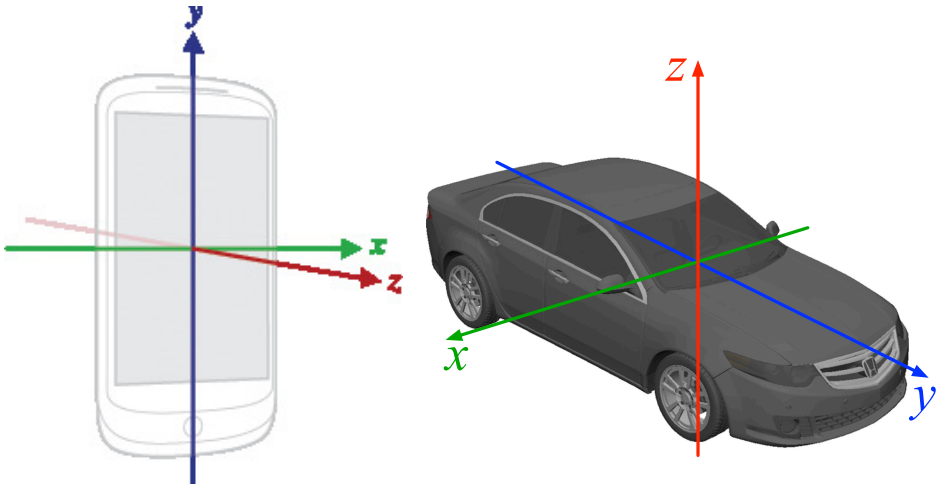
Figure 3.3: When to turn on the GPS module

estimation error could be controlled in a reasonable range. Since the GPS data is what we want to mimic by local sensors, we treat GPS data as groundtruth here. Our target is to see if the acceleration measures got from accelerometer matches the acceleration calculated from GPS data.

First, we give out the definition of the car's and phone's local coordinate systems in Fig. 3.4[1]. Please note that the car's coordinate usually is not aligning with the phone's coordinate system. But for convenience and ease here, our Google Galaxy Nexus is mounted in the car, so the y-axis of the phone aligns with the y-axis of the car. Moreover, the roads we chose are all horizontal. Under these experienments settings, we sampling the GPS data with 1Hz and accelerometer with 50Hz when driving on the road.

After collecting the data, we apply a low-pass filter to remove the fluctuations caused by the intrinsic noisy nature of the sensor and compare the filtered accelerometer data with the acceleration computed from GPS trace in Fig. 3.5[1].

[1] The figure is made by Shaohan Hu



(a) Phone (b) Car

Figure 3.4: Local coordinate systems

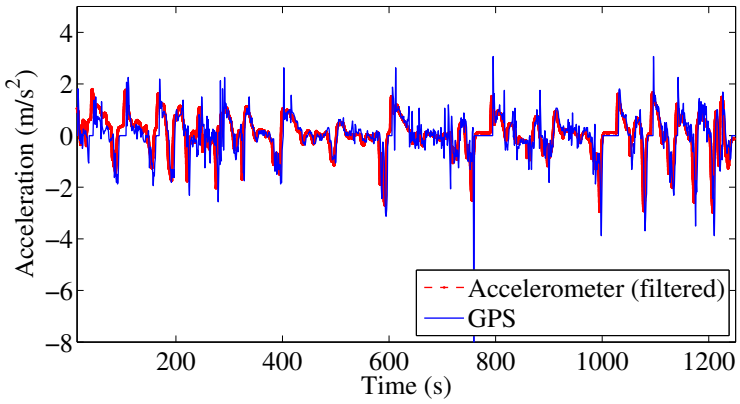
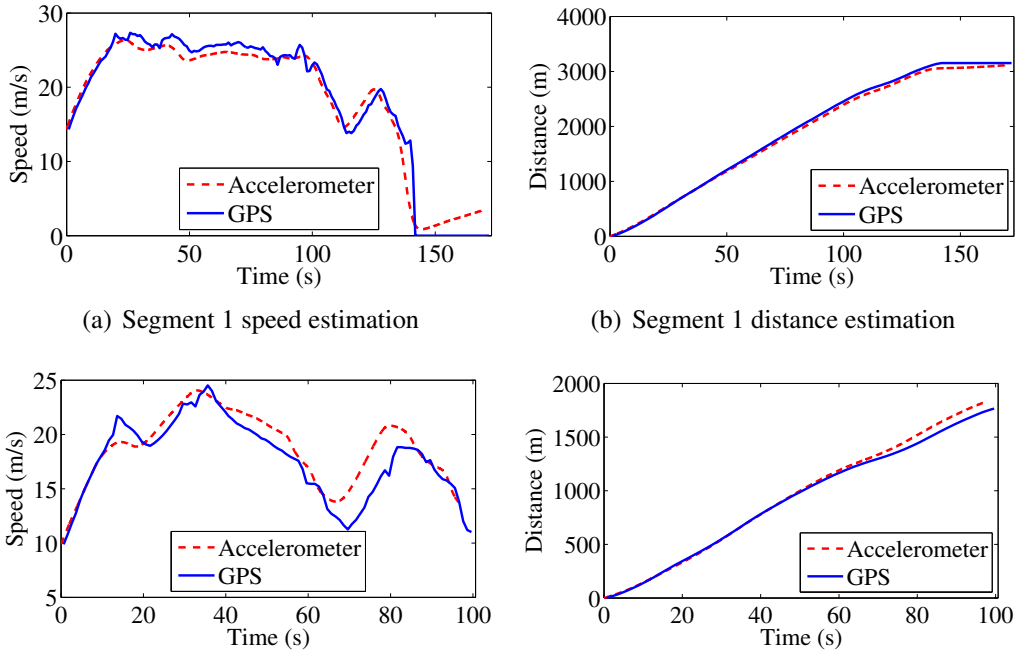


Figure 3.5: Filtered accelerometer data compared to the acceleration computed from GPS trace

Furthermore, we check if the car's distance estimated from integration from the motion matches the real moving distance. Here we show the speed and distance comparision from two segments of the road in Fig. 3.6[1].

To see if the estimation error is under a reasonable range, here we gives the normalized distance estimation error distribution for all the segments, which is distribution of segment estimation error/segment length mathematically, in Fig. 3.7[1]

The feasibility study above support our idea that using local low energy consuming sensors to estimate the location of the car is feasible. In next section, we dig into the car motion estimation in practical situation.



(a) Segment 1 speed estimation

(b) Segment 1 distance estimation

(c) Segment 2 speed estimation

(d) Segment 2 distance estimation

Figure 3.6: Car driving speed and distance estimation using the phone's accelerometer data

3.3 Estimation in Practical

In previous section, we talked about the feasibility study in ideal situation. However, in realistic life, the road can be rugged rather than all horizontal. The phone could be put in any arbitrary position (e.g. on seat, in the car cup-holder or in pocket) in the car, which makes the two coordinate systems in Fig. 3.4 do not align to each other. Even the phone may be put in any arbitrary position in the car, we still assume that during the trip, the driver will not interact with the phone to make the orientation between the phone's coordinate system and the car's coordinate system fluctuate a lot.

Given the situation and assumption above, two problems has come to us before we are able to implement eNav system.

1. Find the transformation between the car's coordination system and the phone's coordination system and map the phone's motion to the car's coordination system to get the car's direction and motion.

2. Since the road may be rugged in realistic life, the noise caused by gravity should be removed

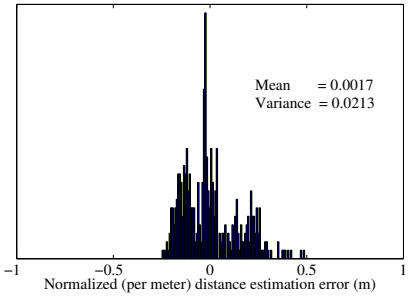


Figure 3.7: Distribution of the normalized distance estimation errors of performing dead-reckoning using the phone's accelerometer data

from the mapping in the first step.

After doing some research, we found that Android originally provided a function called Sensor Fusion [9]. And we also find a solution which is based on Principal Component Analysis (PCA) [10]. In next section, we will introduce both of the methods and why we choose the later one as our final solution.

3.4 From Phone's Coordination System To Car's

3.4.1 Sensor Fusion

Since the phone contains the original method to deal with the problem, we try the sensor fusion first. Sensor fusion is a set of proprietary algorithms implemented by the phone's sensor vendor [11], which basically works as a low-pass filtering on accelerometer and magnetic field sensor readings and high-pass filtering on gyroscope readings. Then sensor fusion combines them to correct the errors of each other and gives a list of *virtual* sensor readings, such as gravity, rotation vector and linear acceleration[12, 9]. The gravity virtual sensor readings means the separation of the earth gravity from the raw accelerometer readings while the linear acceleration represents the phone's actual acceleration's part. The rotation vector contains the rotation matrix, which is what we need to convert between the phone's coordinate system and the earth's global coordinate system (car's coordinate system).

We test the sensor fusion by placing several phones in different positions in a car and collect data

	Mean	Variance
Placement	Fusion	Fusion
Pocket	-0.3437	0.1100
Dashboard	-0.3873	0.3755
Seat	-0.4002	0.0748

Table 3.1: Comparison of the normalized distance estimation error distribution statistics between our proposed PME method *vs* using the phone's proprietary sensor fusion

from fusion virtual sensors and GPS data. Fig. 3.8 shows the comparison between the estimation acceleration and real acceleration got from GPS data in one of the phones, which is put on a rubber pad on the dashboard right below the windshield. We can notice that it is hard to say the estimation value is satisfied to be used for location estimation.

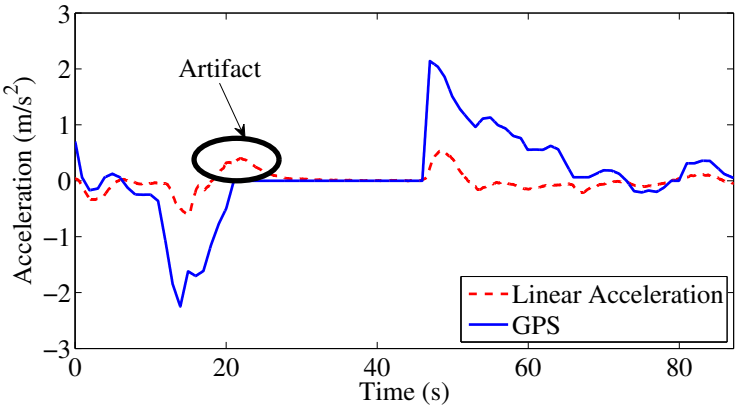


Figure 3.8: Car motion estimation using phone's proprietary Sensor Fusion

Fig. 3.8[1] is the data from one of the phone. Here we gives the normalized distance estimation error histograms on other phones in Fig 3.9[1] and the distribution statistics collected in Table 3.1. The results shows that the dead-reckoning estimations are always under the real distances that the car moved by about 34% to 40% on average, which is a unacceptable range and makes us think sensor fusion is not a appropriate way for us to use in eNav. So we come up with our new methond based on Principal Component Analysis(PCA) [10] in next subsection.

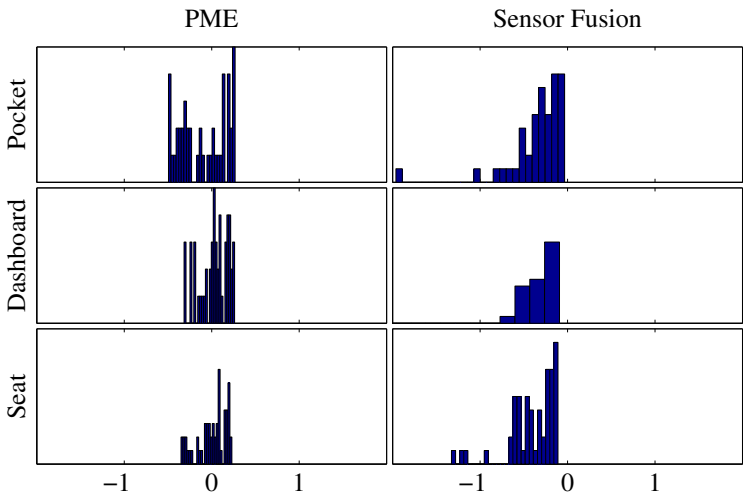


Figure 3.9: Normalized distance estimation error histograms for our proposed PME method *vs* using the phone's proprietary sensor fusion

3.4.2 Principal Motion Estimation

Since the results returned by sensor fusion are not satisfied, we have to give up the simple way and try to figure out a method that can give a better estimation. *Principal Motion Estimation* (PME) is what we called for our new method. We extends the techniques of Principal Component Analysis (PCA) [10] into our Principal Motion Estimation. PCA is usually used to extract an array of components which focus on the variability of data better. Each component is orthogonal to the previous one and will try to capture the remaining variability of the data as much as possibile. In this case, car's moving direction should have the largest acceleration measure. So we can get that value by applying PCA on the phone's 3-axis accelerometer data and extract the first component(i.e., the y-axis in Figure 3.4(b)). If the car are making left/right, this kind of acceleration variability could be captured by the second and third PCA components((x-axis) and vertical movements (z-axis)).

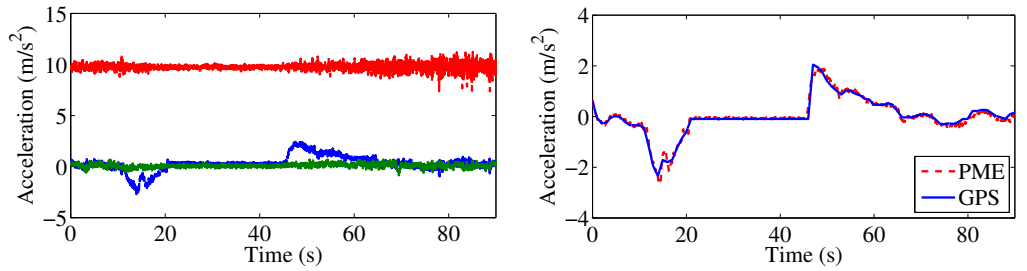
Suppose that we have a set of raw accelerometer data trace, we subtract each value with the mean to remove the gravity's effect on the accelerometer measurements. Then we feed the data into PCA and get a three-column matrix as output. Each column vector is an eigenvector of the input data's covariance matrix. If we multiple the input data matrix with first column vector, we will get the acceleration component on car's moving direction. Here we do not care about the remaining two column vectors since we only want to get the distance that the car has moved. So if $\mathbf{A}$ is the raw data with its mean subtracted and $\mathbf{u}$,$\mathbf{v}$,$\mathbf{w}$ is the output of PCA. $\mathbf{Au}$ is the acceleration

on the direction of car's moving, which is what we want.

We have assumed that the phone's position will not be changed during the trip, which means the transformation angle between the phone's coordinate system and the car's coordinate system will stay the same. And from the mathematical part above, we can notice that the vector $\mathbf{u}$ represents the direction of the car's moving direction. So we only have to calculate the vector $\mathbf{u}$ extracted from the raw accelerometer data once at the beginning of the trip and it can be continously used during the whole trip. But even something unexpected happen to make the phone moved in the car and make the relative orientation of the phone and the car changed, the sensors like acceleromter could detect it (like the way in the "extraneous activities" detection in Jigsaw [13]) and re-run the PCA code to revise the vector $\mathbf{u}$. In the real system, eNav will take the first one minute of the trip as warm-up period to calculate the vector $\mathbf{u}$ and then use it in the following driving. Moreover, the principal component directly produced by the PCA has ambiguous signs, which means it may be in the same or reverse direction. So in the warm-up step, eNav will also sampling the GPS data to adjust this problem. All the details will be covered in Chapter 5.

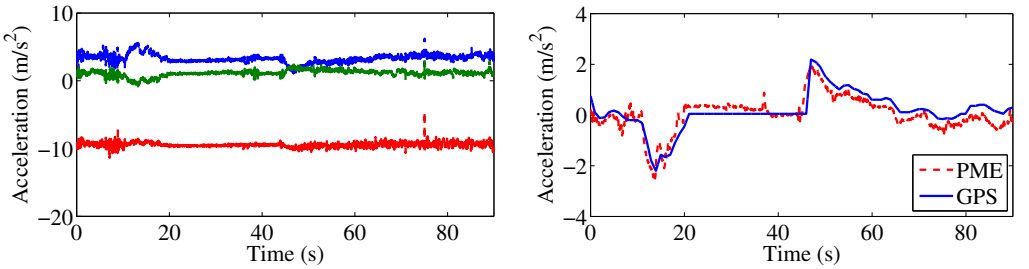
Here we show the experiment results for our Principal Motion Estimation in Fig. 3.10[1]. To compare with results without bias, Fig. 3.10(a), Fig. 3.10(b), Fig. 3.10(c) and Fig. 3.10(d) are processed from data collected from the same driving trips as previous sensor fusion's in section 3.4.1. Fig. 3.10(b) shows the PME results for a relatively ideal situation, in that the phone is on a rubber pad on the dashboard right below the windshield. We can see that the estimation values are almostly in the same line with the groundtruth value in Fig. 3.10(b). Fig. 3.10(d) presents the PME results for the phone placed in the driver's jacket hand pocket. As the driver's body movements, the estimation will be affected by some noises. So we can see that in Fig. 3.10(d), the estimation value does not match groundtruth as well as the value under relatively ideal situation showed in Fig. 3.10(b). Nevertheless, under the same trip, the Principal Motion Estimation performs much better than the sensor fusion provided by the phone.

The experiments before are tested under quite ideal road condition. Here we will try the PME method on the hilly road and show the results in Fig. 3.12[1]. When driving on a hilly road, the ups and downs will generate a non-zero component on the car's driving direction. But before feeding the raw data into PCA, we have make the raw data subtract its mean to reduce the impact caused by the gravity's fluctuation. One part of the trip of Google Map Street is presented in



(a) Raw 3-axis accelerometer data of phone on car dashboard

(b) Estimation result of phone on car dashboard



(c) Raw data of phone in driver's jacket hand pocket

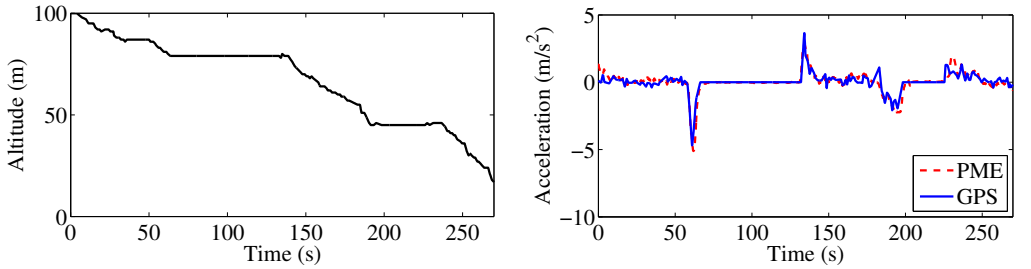
(d) Estimation result of phone in driver's jacket hand pocket

Figure 3.10: Principal Motion Estimation result demonstrations

Fig. 3.11. To show how rough the car had gone through, the altitude trace of the car on the hilly road is drawed in Fig. 3.12(a). However, even under this kind of tough condition, we can see that the PME estimation value still closely catch the groundtruth of the car's acceleration and deceleration motion. Table. 3.2 shows the aggregated distance estimation error distribution statistics. To compare with sensor fusion, we also put the sensor fusion's results in the Table. 3.2. After putting the two results together, it is obvious that the PME method is much better than the sensor fusion in distance estimation for both mean and variance of the normalized errors under all the experiment conditions. So we decided to use the PME in our final eNav system.

In this section, we compared two methods on the estimation of the car's moving distance and decided to use Principal Motion Estimation, which based on Principal Component Analysis as our final solution. In next section, we will describe some mathematical formulations about our estimation model.

Figure 3.11: Google Street View of the hilly road



(a) Altitude trace of car driven on a hilly road (b) Estimation result of car driven on a hilly road

Figure 3.12: Principal Motion Estimation result on hilly Road

3.5 Mathematical Formulation

In this section, we will talk about some mathematical formulations here. But since the mathematics behind dead-reckoning is quite simple, which is just classic Newton's law, we'll go over it quickly.

First, we discretize the whole trip into time slots, which each has a duraion of t. So in i-th slot, we have:

$$\hat{v}_i \quad = \quad \hat{v}_{i-1} + \hat{a}_i t = \hat{v}_{i-1} + a_i t + a_e t.$$

where a_i is the real acceleration acceleration measure for this slot (we take GPS speed readings as groundtruth in our system, so this value is also reading from GPS data). $\hat{a}_i$ is the estimated acceleration reading of the accelerometer. And $(a_e = \hat{a}_i - a_i)$ is the estimation error.

Furthermore, with the formulation about the speed and acceleration, we can the formulation about distance in i-th time slot, which is:

	Mean		Variance	
Placement	PME	Fusion	PME	Fusion
Pocket	-0.0906	-0.3437	0.0685	0.1100
Dashboard	0.0272	-0.3873	0.0269	0.3755
Seat	0.0063	-0.4002	0.0282	0.0748

Table 3.2: Comparison of the normalized distance estimation error distribution statistics between our proposed PME method *vs* using the phone's proprietary sensor fusion

$$\begin{aligned}\hat{s}_i &= \frac{(\hat{v}_{i-1}+\hat{v}_i)\cdot t}{2} = \frac{(v_{i-1}+v_i)\cdot t}{2} + \frac{(v_e^{i-1}+v_e^i)\cdot t}{2}, \\ &= \left[v_0 t + \sum_{k=1}^{i-1} a_k t^2 + \frac{a_i t^2}{2}\right] + \frac{2i-1}{2} a_e t^2 = s_i + s_e^i.\end{aligned}$$

Here we give the assumption that the estimation error a_e is a random varaible following the Normal Distribution with 0 mean and σ as the standard deviation. So we have $a_e \sim \mathcal{N}(0, \sigma^2)$. Furthur, we have $s_e^i \sim \mathcal{N}(0, \sigma_s^2)$, where $\sigma_s = \frac{2i-1}{2}\sigma t^2$.

So in our final system, the eNav will maintain the car's confidence bound ahead of its estimated displacement, i.e., $\hat{s}_i^c = \hat{s}_i + 2\sigma$. With this confidence bound, the eNav system can gurantee that the car has not yet crossed over the next way-point with confidence. Then, if our estimated location with confidence bound is close enough to the turning point, the eNav will need to get the accurate location of the car by sampling the GPS data, and see if the car is actually close to the way-point or not, and act accordingly.

Specifically, the bound $\hat{s}_i^c$ gives us a confidence measure of 97.72%, as shown below,

$$\mathbf{P}(s_e^i \geq -2\sigma_s) = \mathbf{P}(v_e^i \geq -2\sigma_v) = \mathbf{P}(a_e \geq -2\sigma) = 0.9772.$$

And in the implementation of eNav, the standard deviation σ for the acceleration estimation errors will be obtained during the initial warm-up step of the navigation. Fig. 3.13[1] shows the standard deviation on acceleration estimation errors under various phone placement scenarios we

mentioned above in previous section. As the results show, the standard deviation on acceleration estimation errors have drawn the same conclusion as in Fig. 3.10, which the driver's pocket position shows the worst, whereas the one placed on the car's dashboard gives the best.

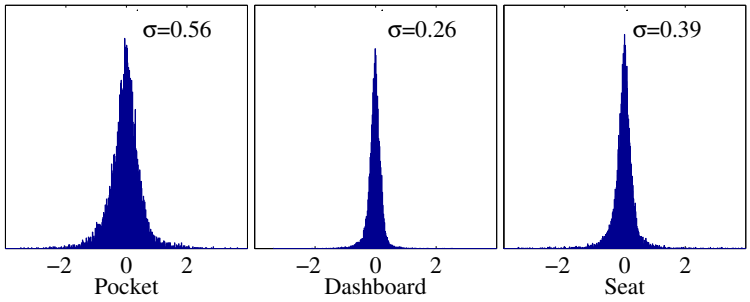


Figure 3.13: Acceleration estimation error distributions

One more thing we want to mention here is that we also consider the speed limitation in our final eNav system, which is meaningful in real life and help us to limit the estimation error. With the speed limitation, the accumulative error will be bounded in a range and avoiding the overestimation caused by snowball effect.

In this section, we model our estimation method into mathematical way and discuss the confidence about estimation error used in our eNav system. In next section, we will talk about two more plug-in modules for enhancements in our eNav system to make it work better.

3.6 Plug-ins For Enhancements

In this section, we will talk about two enhancements module in our eNav system, which we use to improve the location estimation for navigation quality. Here we just cover how why each of them works as an enhancement and how it is done. The implementation details will be covered in Chapter 5.

3.6.1 Car Idle Detection

We have discussed that how it is hard to estimate the car's motion by using acceleration since it is a double integration problem, which will bring error in the solution. We have to control the error

Classification Algorithm	Car Idle (%)
Decision Tree	99.80
Support Vector Machine	96.35
Naive Bayes	98.17

Table 3.3: Car idling detection accuracy comparisons using various classification algorithms (10-fold cross-validation)

within a controlled range. However, to detect whether an object is moving or not is much easier since it is what acceleration originally represents. There is still a gap between the answer and us, which is that the phone's motion does not equal to the car's motion. Even the car is keeping idle, the phone may still sensor fluctuations caused by the car's engine, noise from enviroment and noise caused by sensor itself. Since what we need is a yes or no status, we can treat the car idle detection as a binary classifier and formulate it as a classic classification problem. We take statistical measuresas like $mean$, min, max and $standard\ deviation$ our features to train our threshold for the classifier.

There are various classification algorithms, so we did some experiments to get the best one for our specific problem. We use 10-fold cross-validation to compare their accuracies (the average percentage of correctly labeled time slots among all slots tested on). Here we also use the GPS data as our groundtruth and to label the data. Our experiment result shows that the decision trees [14] consistently outperforms other algorithms and achieves classification accuracies well around 99%, as show in Table 3.3. Therefore, we choose to use decision trees in our final system design.

With the car idle detection described above, the eNav will reset both the acceleration and estimated speed. Therefore, we can further prevent the estimation error from growing erroneously.

3.6.2 Car Turning Detection

Another specific situation that we can get the car's location without using GPS data is when the car is make turning at crossroad. Combined with the road intersection information from map, we can probably get the car's accurate location. The question here is similar to the idle detection problem, so we also treat it as a binary classifier. Here we firstly try the data from magnitude trace of the

Classification Algorithm	Car Turning (%)
Decision Tree	98.89
Support Vector Machine	69.48
Naive Bayes	63.63

Table 3.4: Car turning detection accuracy comparisons using various classification algorithms (10-fold cross-validation)

gyroscope readings and the the decision tree gives great result, but according to our energy profiling, gyroscope's energy consumption rate (about 0.0214W) is two orders of magnitude greater than that of the accelerometer (about 0.000488W). So we try it with accelerometer data and it also gives similarly near performance, which makes us decide to use accelerometer data in our final solution. And statistical measures such as $mean$, min, max and $standard\ deviation$ are still using here as features to train the turning classifier and GPS data are still works as groundtruth and used to label data. 10-fold cross-validation is used again to compare different algorithms. Table. 3.4 shows that decision trees wins in this contest. Therefore, we decide to use the decision trees in both of the idle detection and turning detection case in our eNav, which achieves around 99% accuracy.

In next chapter, we will cover the navigation flow of eNav to give a general idea about how it really works.

CHAPTER 4

NAVIGATION FLOW

Before we get into the implementation details, this chapter is gonna discuss about the navigation flow to show how the whole eNav system works during driving and combines the components talked about above together to give a global view of the system.

The navigation flow is illustrated in Fig. 4.1. Nodes marked as Ox's and Dx's correspond to the basic operations and decisions in the navigation flow. Nodes marked with eDx's correspond to the enhancements.

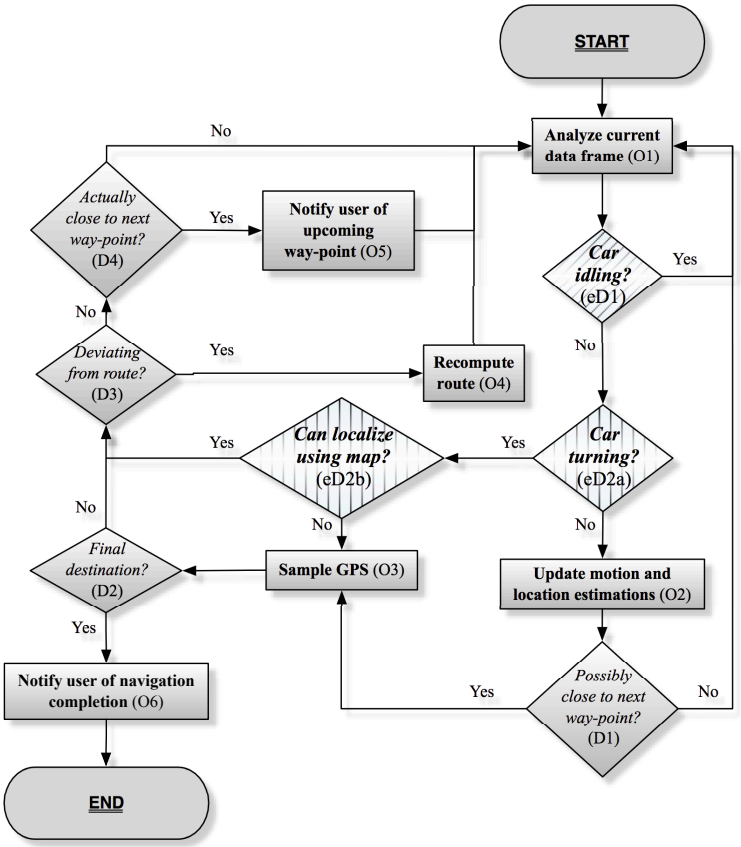


Figure 4.1: eNav's Navigation Flow

At the beginning of the route (the first minute), which we called warm-up phrase in previous chapters, the GPS module is turned on with the acceleromter. The data collected in this step is used to initialize some parameters mentioned above like the standard deviation of acceleration estimation error, the PME principal component vector and its sign. After that, the normal navigation starts.

Below, we will discuss the flow based on the "O" nodes.

1. O1 will generate the car's principal motion when it get new data frame from the sensor module. Then it will check if the car is idling. If the car is idling, go back to O1. If the car is not idling and it is turning, eNav will try to localize the car using map information. And if it cannot localize the location by map, it will go to O3. If the car is neither idling nor turning, the flow goes to O2.

2. O2 is responsible for update motion and location estimations. If it is still far away from the next way-point, it will go to O1. If it is possibly close to the next way-point, it go to O3. Here *"possibly"* means the estimation of the least amount of time it takes to reach the next way-point is under a threshold, which is calculated by the estimated speed and the displacement confidence bound ($\hat{s}_i^c = \hat{s}_i + 2\sigma$ discussed before.

3. O3 will start the GPS module and sample GPS data. And after eNav get the accurate location of the car, regardless of whether it is from using map or GPS sampling, it will first check if the car has reached the destination. If here is the final destination, it will go to O6. If not, it will check if the driver has made some mistake and is deviating from the route. The flow will go to O4 if the car is deviating from the route that eNav gives. If everything works fine and the car is not close to the next way-point enough, go back to O1. Otherwise, go to O5.

4. If the flow goes into O4, there might be something that should not have happened. The car has been deviating from the original route that eNav gives to the driver, so eNav will recompute the route from the car's current location to the final destination and restart the navigation.

5. O5 will play a voice message to notify the driver that the next way-point is upcoming and allow the driver to have time to change the lane and prepare for the turning. The voice

message will contain the information about how far the next way-point is and make the driver to pay attention to the guide board on the road. After that, the flow will go to O1.

6. O6 is the state which will only be showed when the car arrived the destination. It will play a voice message to let the driver know that he or she has been reached the final destination.

In this chapter, we go over the whole flow of how eNav works. And in next chapter, we will go into more details about the how everything is implemented.

CHAPTER 5

IMPLEMENTATION

In this chapter, we discuss the implementation details and engineer design of our eNav prototype system.

5.1 Prototype Navigation App

Our eNav prototype app is implemented on Google's Galaxy Nexus Android phones [15]. We start with the codes from scratch and the route information is obtained by using Google Maps API [16]. When a user enables the energy-saving mode in navigation, GPS sampling becomes selective and car localization becomes inaccurate between way-points. This is hidden from the user because the screen will remain off and navigation will be based on deliveries of voice notifications to the user. The user is never exposed to inaccurate positions during navigation. If the user press a phone button (e.g. volume button) at any time, GPS sampling restarts and accurate positioning is restored, effectively exiting the energy saving mode.

5.1.1 Main Components

Fig. 5.1 shows the main components of the prototype app. When the driver open eNav, he or she will interact with the *Interface Activity*. The interface is presented in Fig.5.2. After the driver put in the origin, destination and the options, the *Map Activity* will be activated. The *Screen On GPS On* option is for test usage. If *Screen On GPS On* is checked, when the user turn on the screen, the GPS will be activitated and the estimated location will not be discovered by driver.

The Map Activity is used for rendering and drawing the map. It will first start a thread to ask for the route information from Google Maps API and wait until get it. Then it starts to draw the route

on the map. After that, it will register GPS module and local sensors to the system with the callback functions and start the main loop thread. The red rectangle in Fig. 5.1 is the global variables that all the threads maintain and update together. The GPS module (if activited) and the local sensors will call the given callback functions to update the information in the red rectangle. The main loop thread will check the global flags and variables periodically and do the according operations like activate/deactivate GPS module, notify the user about next way-point, final destination or deviation, etc. Fig.5.3 and Fig.5.4 shows the Map Activity when eNav is working under driving (the red point is the real location and the green point is the estimaed location of the car). Fig.5.3 is shows the situation when eNav is under warm-up phrase or the screen is on, the driver can only see the real location. For illustration, I did not check the *Screen On GPS On*, so we can see that in Fig.5.4, the estimation point (green) is in advance of the real location (red).

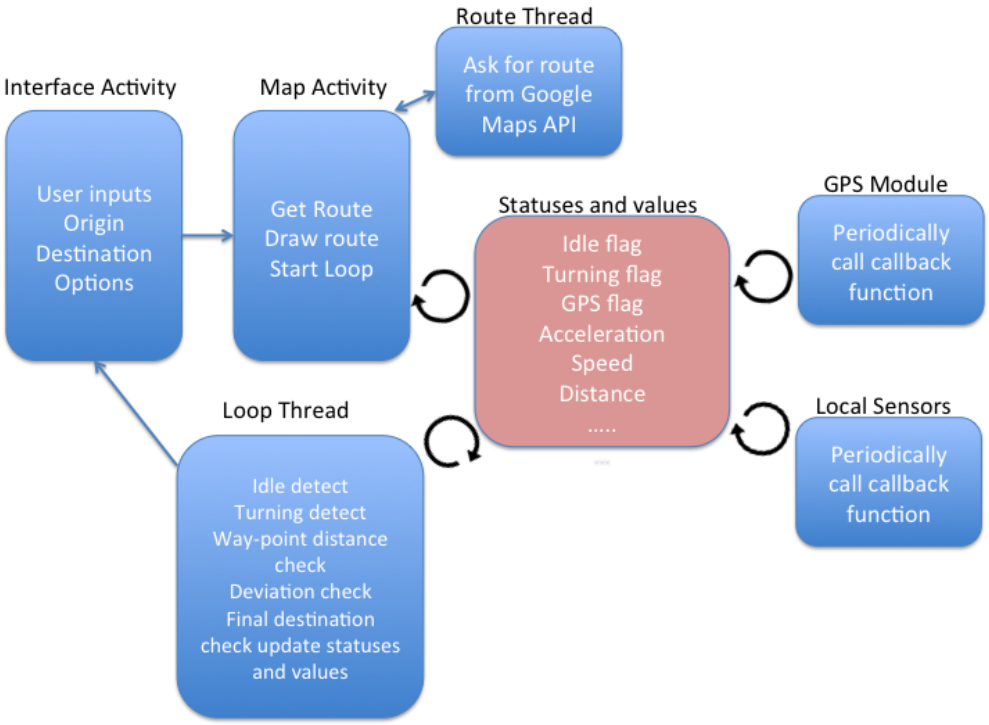


Figure 5.1: Main Components of eNav Prototype App

Moreover, the idle and turning detection mentioned as enhancements are also conducted in the main loop thread. eNav will have a buffer of fixed size (queue) to store the most recent data for past a duration of peroid (e.g one minute). The statistical measures like $mean$, min, max and

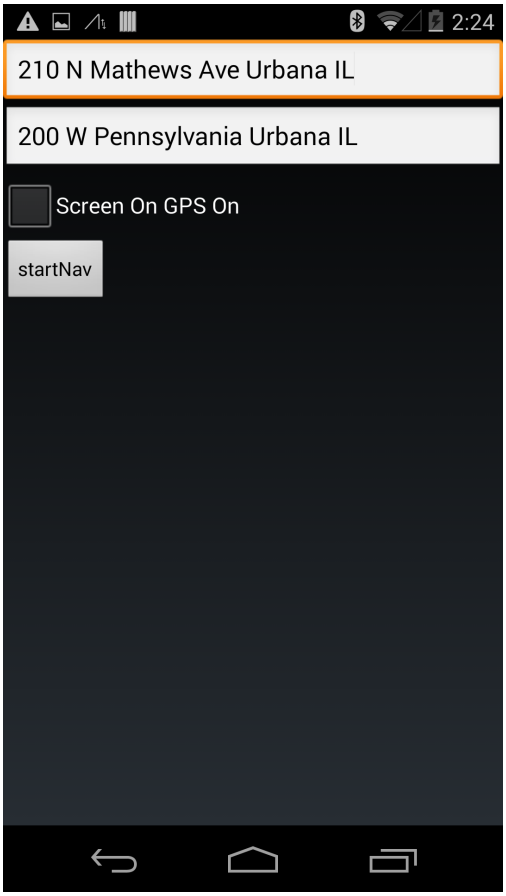


Figure 5.2: User Interface of eNav

$variance$ of the data in the buffer are also updated in the loop thread. So eNav can do the idle and turning check in real time.

User Interface Activity

In our prototype application, the user interface activity only provides the basic information input textarea like origin, destination. After the user push the start button, an intent will be generated and sent to start the Map Activity with the information that the user has input in.

Map Activity

The Map Activity works as the map interface in the navigation. We use Google Maps API to render the map since Google Maps is the most popular map service online and most of people are

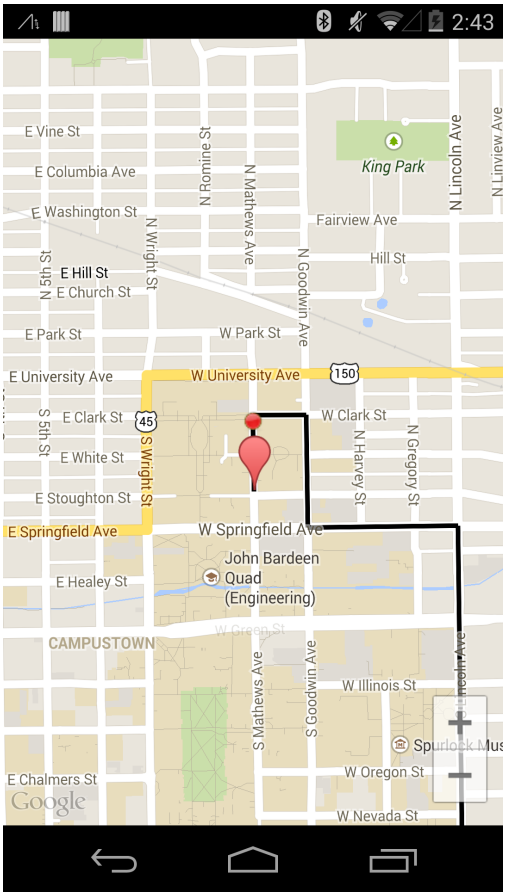


Figure 5.3: Example of Warm-up Phrase or when the Screen is On

familar with it. Most of the time, eNav conducts the navigation by playing voice messages and the screen is off. But when the driver want to turn on the screen, the GPS module will be activited immediately to show the real GPS location where the car is. By this way, the user can feel like he is just using a traditional GPS navigator and will not be confused by the estimated location. The estimated location usually overestimates the car's motion distance, thus the location will be furthur than where is the car actually is. This kind of information could have negative influences on driver's judgement and may cause tension during driving, which is undesirable when someone is driving in a car. So we decided to cover the information from the users.

Main Loop Thread

The main loop thread works as the core component of the eNav application. It is implemented as a timer task, which will periodically call the check function provided by *eNavigation* class. The

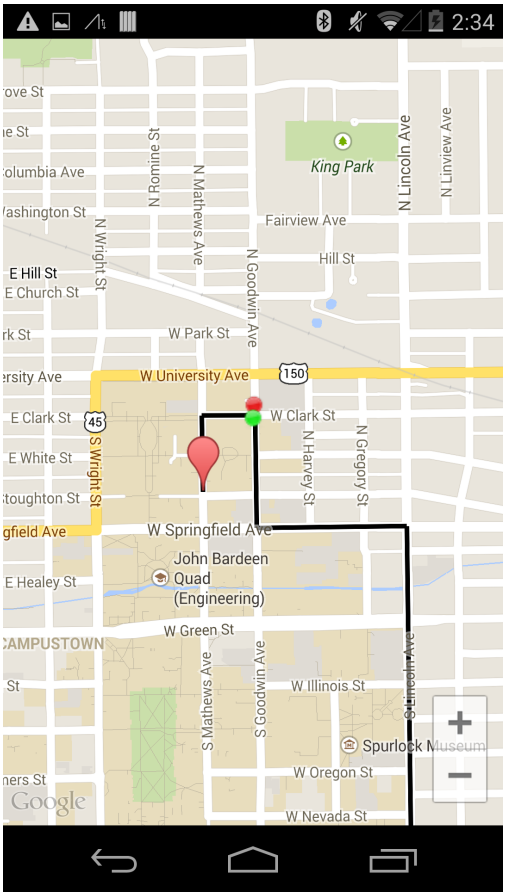


Figure 5.4: Example of the Estimation Location and the Real Location

eNavigation class extends from *NavigationBase* class. The check function will execute the idle detection, turning detection, way-point distance check, deviation check, final destination check, update the car's motion distance and update the statuses and flags according to the results of the checks and detection done above. For the idle and turning detection, eNav will use the statistical measures of the data in the buffer and check them with the two classifiers. Since the local sensor's frequency is very high, here the buffer is backed up first and then do the statistical measures calculation in order not to block the sensors to feed the data. For the way-point distance, if the car's estimated arriving time is under the threshold, the eNav will start the GPS to get the sampling and reset everything. However, the GPS module need a period of time to warm-up (GPS sampling is not stable before warm-up). So the eNav will start GPS in advance and then use the data from the GPS samplings. If the location from GPS sampling shows that the car is still not close enough to the next way-point, the GPS module will be deactivated again for energy-saving. If the final

destination is detected, eNav will play the notice message to the driver and do the clean up work to ready to go back to the user interface.

Shared Global Variables

The shared global variables are the values and flags such as the estimation value of distance, speed, GPS module on/off, buffer for sensors' data, statistical measures for the data, estimated distance to the next way-point and etc. All these data are shared by multiple threads. So the mutex and semaphore are used here to prevent the data collision. However, the frequency of the local sensors could be very high so that the usage of mutex and semaphore may cause the block of the thread. And this kind of block could lead to more and more blocks, which will lead the application to collapse. In order to solve that problem, techniques like double buffer and statistical measures's features are exploited here to avoid long *for* loop.

5.1.2 Data Feed and Simulation

For convenience in debug and experiments, the eNav application supports both the real driving mode and simulation mode. We use the acceleration and GPS sampling collected from other phones to feed the data to eNav to make it think that this is a real driving trip and check what happened during the trip.

5.1.3 Battery Status

For evalution, the eNav also contains the component to log the battery status of the smartphone at the end of each trip. When the driver starts the navigation and arrives the final destination, the status of the battery of the phone will be logged into the local file into the external storage media for analysis. Due the limitation of Android API, the battery information component only logs the percentage number of the battery volume in the current prototype version. In the future, I plan to log more details information about the battery in real time by hacking into the native development so that we can analysis the battery usage during the trip and try to improve our eNav application.

5.1.4 Road Speed Limit

About the road speed limit information we mentioned above, we found that popular GPS navigation hardware manufacturers [17, 18] have provided the road speed limit information during navigation within their products. So for the developers who may want to implement the idea of eNav in their own products, the road speed limit will not be a big problem to achieve. And in our eNav prototype, we use the data from Wikispeedia [19], which is a web service that provide the publicly speed limit data collected by crowd-source.

5.1.5 Road Intersection Information

We try to find some available road intersection location services provider or data set but find nothing. Thus we decided to use the data from OpenStreetMap [20] and get what we need after processing. An intersection is a location that appears in two or more roads. In OpenStreetMap, a pinpoint is represented as *node*, which is a single geospatial point. Fig. 5.5 shows an example of node from OpenStreetMap Wiki[21]. While a road is represented as *way*, which contains the sequence of nodes that are in it. Fig. 5.6 is an example of way from OpenStreetMap Wiki[21] Therefore, we can find the intersection information by looking for the common nodes within multiple ways. The intersection information data is extracted offline and stored on the phone locally for realtime use for navigation.

```
<node id="25496583" lat="51.5173639" lon="-0.140043" version="1" changeset="203496" user="80n"
    <tag k="highway" v="traffic_signals"/>
</node>
```

Figure 5.5: Example of Node data structure in OpenStreetMap

5.2 Future Work

The eNav application now is just a prototype for our evaluation and experiment, which is far away from a product that can be published. In the future, we plan to do a lot of improvements to make it more robust and high-quality. And the interface now is a bit simple and not as convienient as popu-

```
<way id="5090250" visible="true" timestamp="2009-01-19T19:07:25Z" version="8"
  <nd ref="822403"/>
  <nd ref="21533912"/>
  <nd ref="821601"/>
  <nd ref="21533910"/>
  <nd ref="135791608"/>
  <nd ref="333725784"/>
  <nd ref="333725781"/>
  <nd ref="333725774"/>
  <nd ref="333725776"/>
  <nd ref="823771"/>
  <tag k="highway" v="residential"/>
  <tag k="name" v="Clipstone Street"/>
  <tag k="oneway" v="yes"/>
</way>
```

Figure 5.6: Example of Way data structure in OpenStreetMap

lar map application like Google Maps[22], Yahoo Maps[23], Bing Maps[24], OpenStreetMap[20] and IOS maps[25]. In the future, we plan to make it more friendly and try to provide the same experience as good as the popular maps services.

CHAPTER 6

EVALUATION

In this chapter, we talk about how we evalute the eNav system based on energy consumption. Here we just show the basic model and the results. For more details, please read Shaohan Hu's paper[8].

6.1 Energy Model

In this section, we will build a model for us to calculate the energy-saving for the GPS module and the local sensors. Fig. 6.1 shows the multimeter we use to log the energy consuming.

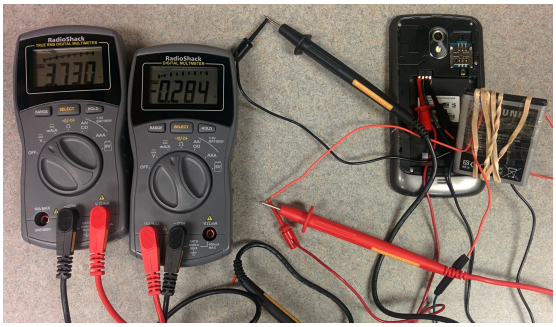


Figure 6.1: Configuration of phone energy consumption measurements. The left multimeter is measuring the voltage, and right one current.

We logged the battery status of the phone from full to 50% level with sampling the local sensors' data at 10Hz. We get that the accelerometer's power is 0.0488mW and the gyroscope's power is 2.14mW. For the GPS module, we logged the values by different sampling rate (from 1Hz to 0.033Hz). Fig. 6.2[1] shows the phone's GPS module energy consumption measurements with increasing sampling period. And we get the fitted model by regression as below.

$$E = -2.4365 \times e^{-0.058383 \times T} + 2.4462,$$

where E is the amount of energy (J) consumed by the current GPS sample and T the time (s) elapse since the previous one.

Fig. 6.3[1] shows phone's GPS energy consumption trends with increasing sampling period on per-sample energy vs total energy for 1min of sampling. We can see that it will consume less energy if the sampling rate is higher under a fixed sampling duration.

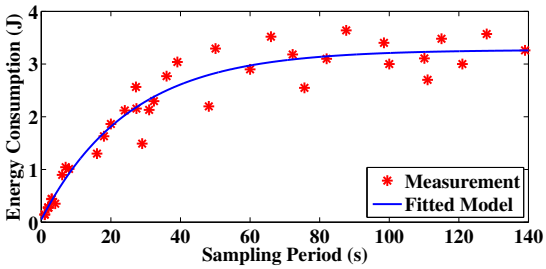


Figure 6.2: Phone's GPS module energy consumption measurements and the fitted model

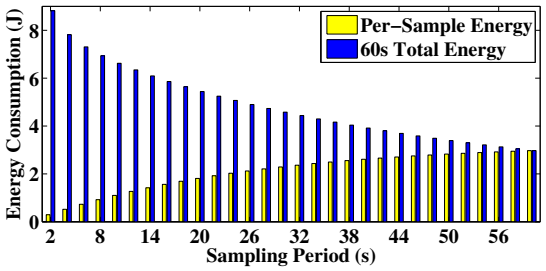


Figure 6.3: Phone's GPS energy consumption trends with increasing sampling period: per-sample energy vs total energy for 1min of sampling

6.2 User Study

We conduct a navigation user study on 33 external volunteer participants from various background (different departments of the university, of both genders, ages ranging from 20s to 40s). The participants are distributed the phone with our eNav prototype application. They are told that the eNav was designed to navigation by voice message and they can put the phone wherever they would like to place. Moreover, even though the default setting of eNav is screen off and guiding by voice message, they could turn on the screen any time during the trip if they want to check their location. A total of over 2000 miles of trip were logged under various conditions, such as

	Energy Savings (%)
eNav with Idle & Turn Detections	78.37
eNav with Turn Detection	73.42
eNav with Idle Detection	70.59
Basic eNav	65.64

Table 6.1: Energy savings of various navigation schemes as compared to the baseline navigation strategy

urban, rural; rush hour, non rush hour; daytime, nighttime; sunny, rainy, snowy. Users were given randomly and unfamilar desitnation so that they have to rely on the instructions of the navigation application without the noise from their previous knowledgement about the roads.

6.3 Energy Saving and Navigation Quality

In the whole study, all the participants are OK with the voice instructions and most of them only turned on the screen once or twice (except one who is a new driver that just got her licence). And no one has ever missed a way-point during the trip.

We set our sampling GPS at 1Hz and turn on the phone screen only when the car is close to the way-point. Based on this setting, we compared the different schemes of eNav system in Table. 6.1. The basic scheme is the one without the two plugins for enhancement. As we can see that, basic eNav could save energy by about 55%, and 65% with the two plugins. And if a user would like to drive only by relying on voice messages, the result could achieve 82% of energy saving.

Fig. 6.4[1] shows the energy saving distributions over different road segment lengths. And we can notice that the longer the segment is, the more energy that eNav can save, which is an expected results. If the route is long and the way-points are limited as we talked before, there must be several segments with large length. So that the time that GPS module is off is longer, which saves more energy.

Fig. 6.5[1] shows a complete navigation example for an entire trip using eNav with both car-idle and turn detection modes enabled. Every single segment's initial displacement value is set to 0 for ease and clarity of illustration. The trip contains both long segments and short segments. We can see that how eNav was working directly in Fig. 6.5 and eNav seldomly sampled the GPS data.

Moreover, idle and turing detection could also be seen in the figure (I_a, I_b, and I_c).

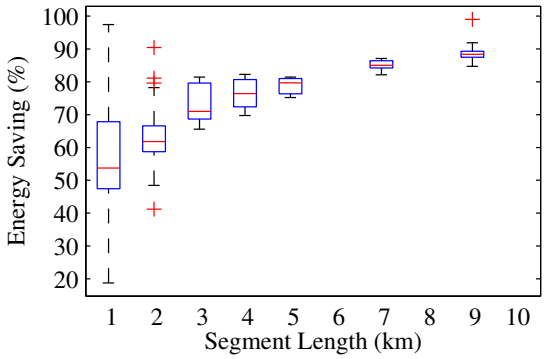


Figure 6.4: Energy saving distributions over different road segment lengths (the edges of the box are the 25th and 75th percentiles; the whiskers extend to the most extremes; outliers are individually marked)

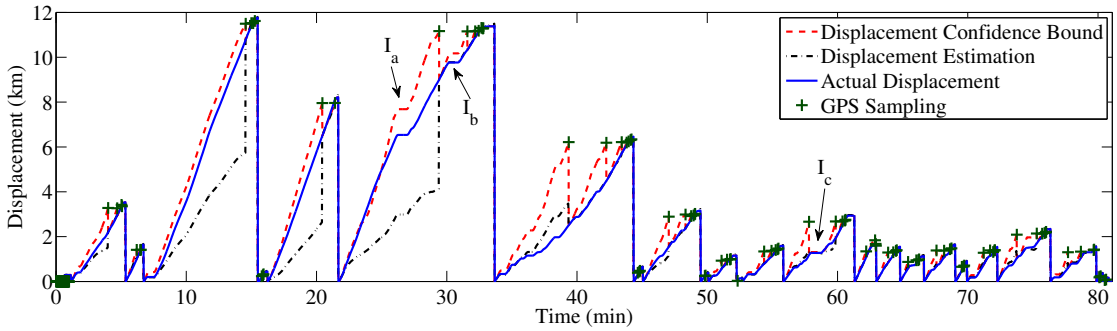


Figure 6.5: A complete navigation session for an entire trip using eNav with both car-idle and turn detection modes enabled.

CHAPTER 7

GREENGPS

In this chapter, we will talk about an extension work about the combination between GreenGPS[1] and our eNav.

7.1 GreenGPS

Traditional map service like Google Maps and OpenStreetMap will usually return the fastest or shortest route to the user. While GreenGPS is a novel service that gives the most fuel-efficient route, which is called "green" route and it consumes the least amount of fuel. GreenGPS exploits the data in On Board Diagnostic-II interface on the car, which became a standard for cars produced in United States after 1996. The study in paper[1] shows that user could save about 10% of the fuel by using GreenGPS service, which reduces the CO2 emissions and saves money. Fig. 7.1 shows how the GreenGPS web service looks like. User have to input the car's maker, model, year and class besides the origin and destination. After that, the GreenGPS will give you the routes look like in Fig. 7.2. The red, blue and green lines repesents shortest, fastest and green route respectively.

7.2 Combination

Our eNav aims to design an appplication that saves energy of phone, while GreenGPS is to save the energy of vehicle. If we combine them together, it could provide a service that can help drivers to use the least energy to finish a trip.

With the idea to provide least energy trip service, I develop a open API for GreenGPS that other developers can use. Due to compatibility, the API mimics the format of Google Maps API. For now, the API only supports json format, which is showed in 7.3. To get a better view of the API, I

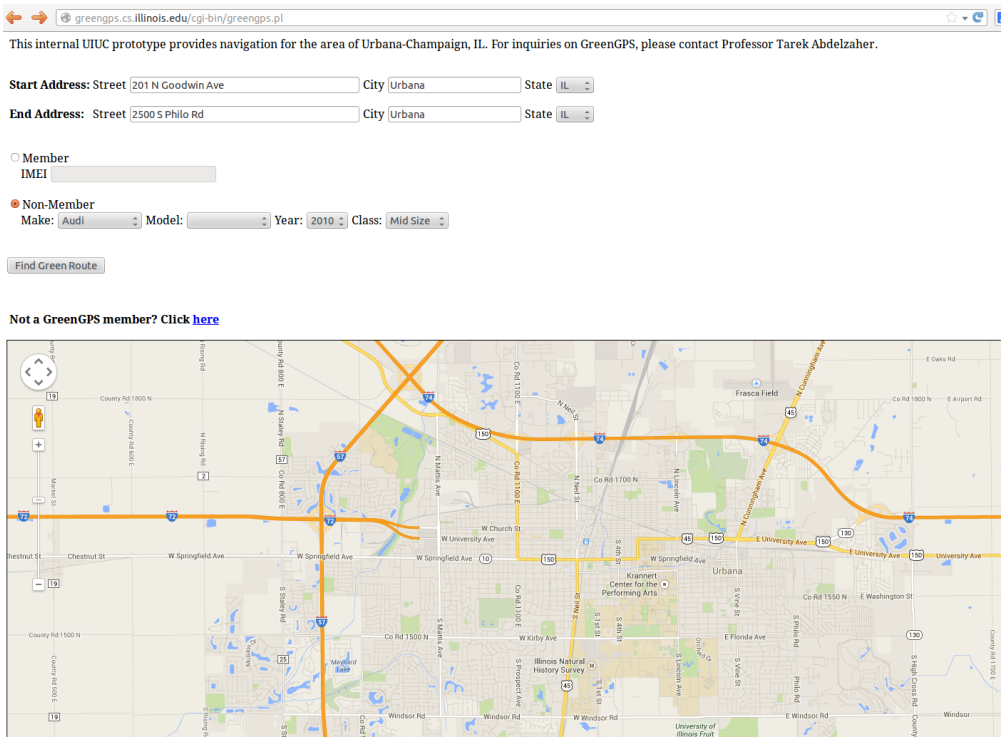


Figure 7.1: GreenGPS web interface

put the results in an online json viewer[26], which is presented in 7.4.

7.3 Problem

Our eNav application demands the voice instructions to guide the driver when the screen is off. Google Maps API contains that kind of information while GreenGPS do not have originally. So we have to develop a system that generates the voice instructions for our green route and it is under developing.

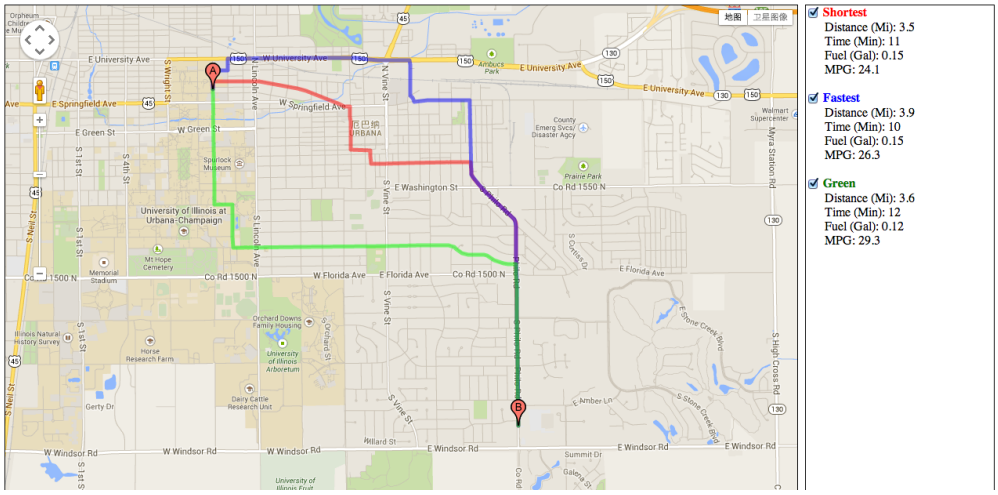


Figure 7.2: Routes that GreenGPS returns

Figure 7.3: GreenGPS API

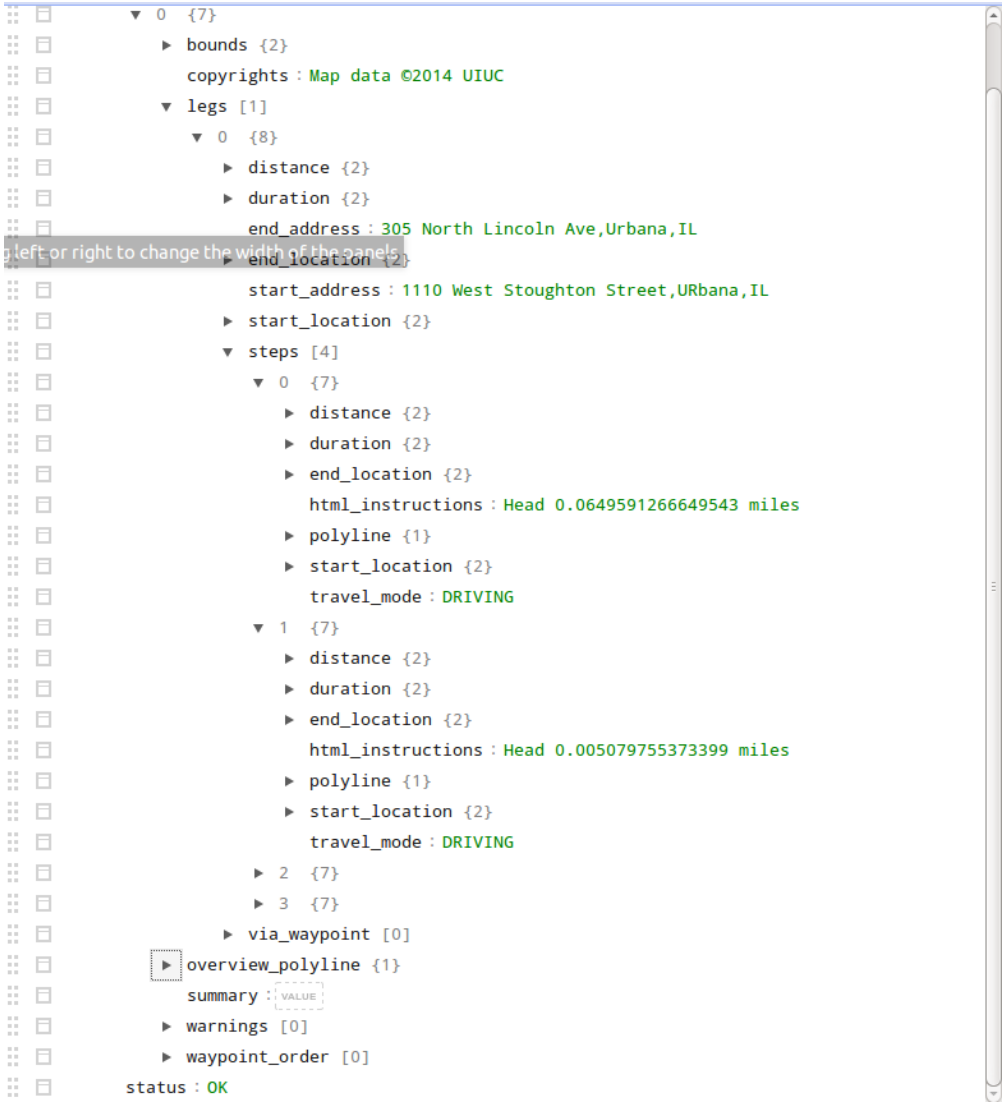


Figure 7.4: GreenGPS API in JSON Viewer

CHAPTER 8

RELATED WORK

Smartphone is more popular in working as navigator in our life, which leads to a lot of studies that focuses on smartphone-based outdoor location sensing [27, 28, 29, 30, 13, 31, 32, 33] and indoor location sensing for pedestrian[27, 34].

However, the energy-consuming GPS module [35, 13, 36, 31, 37] makes the location sensing quality and the energy saving a conflict. Works like static [31] or adaptive duty cycle [36, 33, 38], using local sensors' to control GPS module[32, 13], network-based methods[36] and multi-modal-based approaches[34, 34, 39] are trying to find the balance of the trade-off.

Our eNav are trying to sacrifice the useless location sensing in the middle of the segment to gurantee both the navigation quality and energy-saving.

CHAPTER 9

CONCLUSION

In this thesis, we introduce eNav, an smartphone-based energy-efficient vehicular navigation system in both theory and implementation details. Our eNav sacrifices the useless location information during the segment and only cares about the accurate location around the way-point. This kind of design helps us to achieve both of the navigation quality and energy-saving. As the results shows, eNav achieves over 80% of the energy during the trip and no way-point is missing in our user study.

APPENDIX A

SURVEY QUESTIONS

Questions	**Choices**
Age?	
Gender?	a) Male b) Female
How often do you drive a car?	a) Very rarely b) Once or twice a week c) Everyday
Why do you drive? (Check all that apply)	a) To go to school b) To go to work c) To go shopping d) For occasional entertainment e) For long distance travel
How long is your average drive?	a) 15 minutes or less b) About half an hour c) About one hour d) More than an hour
Did you ever use a GPS navigation system in your car?	a) Yes d) No
Do you keep a phone charger or adaptor to enable charging your phone while in the car?	a) Yes, always b) Most of the time c) Sometimes d) Very rarely
Did you ever use a phone-based GPS navigation system in your car (or a rental car)?	a) Yes b) No

If you answered Yes to either of the above, do you usually plug-in your phone to the car charger when using phone navigation in a car?	a) No b) Sometimes c) Most of the time d) Yes, always
Did you ever run out of battery on the phone while using the phone for navigation?	a) Yes b) No
Imagine a phone-based GPS navigation system with an optional battery-saving mode that turns off the screen when it is not needed (e.g., when you are not moving or far from your next turn-point). Voice navigation will continue at all times. How useful would you find this battery saving feature?	a) I would not find it useful b) I'd use it when my battery is low c) I'd make it the default mode for my navigator
Which do you think is more important for vehicular GPS navigation: screen display, or turn-by-turn voice guidance?	a) Screen display b) Voice guidance c) Either
If during navigation your phone battery is running low, and turning off your screen can save significant amount of battery life, would you be willing to do turn off the screen and rely just on turn-by-turn voice guidance?	a) Yes b) No

Table A.1: Survey questions and responses.

REFERENCES

[1] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher, "Greengps: a participatory sensing fuel-efficient maps application," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 151–164.

[2] Electronista, "Garmin profits drop as hardware sales continue decline http://www.electronista.com/articles/13/02/20/internal.forecast.expects.troubles.to.continue/," 2013.

[3] Andrew Lipsman, Carmela Aquino , "2013 mobile future in focus," 2013.

[4] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, 2010, pp. 21–21.

[5] Auto Rental News, "2012 u.s. car rental market fleet, locations and revenue http://www.autorentalnews.com/fileviewer/1650.aspx," 2012.

[6] "Crowdflower.com," http://www.crowdflower.com/.

[7] C. R. Gallistel, *The organization of learning*. MIT press, 1990.

[8] S. Hu, L. Su, S. Li, S. Wang, C. Pan, S. Gu, M. T. A. Amin, H. Liu, S. Nath, R. R. Choudhury, and T. F. Abdelzaher, "enav: Smartphone-based energy efficient location sensing for low-power vehicular navigation," Department of Computer Science, University of Illinois at Urbana-Champaign, Tech. Rep. [Online]. Available: https://www.ideals.illinois.edu/handle/2142/48917

[9] G. Milette and A. Stroud, *Professional Android Sensor Programming*. Wrox, 2012.

[10] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2005.

[11] InvenSense, "http://www.invensense.com/," 2013.

[12] Android Developer Reference, "http://developer.android.com/reference," 2013.

[13] H. Lu, J. Yang, Z. Liu, N. Lane, T. Choudhury, and A. Campbell, "The jigsaw continuous sensing engine for mobile phone applications," in *SenSys*, 2010.

[14] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[15] Galaxy Nexus, "https://play.google.com/store/devices/details/Galaxy_Nexus_HSPA?id=galaxy_nexus_hspa," 2013.

[16] "Google maps api," https://developers.google.com/maps/.

[17] Garmin, "http://www.garmin.com/," 2013.

[18] TomTom, "http://www.tomtom.com/," 2013.

[19] Wikispeedia, "http://www.wikispeedia.org/," 2013.

[20] OpenStreetMap, "http://www.openstreetmap.org/," 2014.

[21] "Openstreetmap wiki," http://wiki.openstreetmap.org/.

[22] G. Maps, "https://www.google.com/mobile/maps/," 2014.

[23] Y. Maps, "http://maps.yahoo.com/," 2014.

[24] B. Maps, "http://apps.microsoft.com/windows/en-us/app/maps/97a2179c-38be-45a3-933e-0d2dbf14a142," 2014.

[25] I. Maps, "http://www.apple.com/ios/maps/," 2014.

[26] "Json editor online," http://www.jsoneditoronline.org/.

[27] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury, "Did you see bob?: human localization using mobile phones," in *MobiCom*, 2010.
[28] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 2, p. 13, 2010.
[29] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and L. Cox, "Enloc: Energy-efficient localization for mobile phones," in *INFOCOM*, 2009.
[30] M. Azizyan, I. Constandache, and R. Roy Choudhury, "Surroundsense: mobile phone localization via ambience fingerprinting," in *MobiCom*, 2009.
[31] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application," in *SenSys*, 2008.
[32] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *SenSys*, 2008.
[33] J. Paek, J. Kim, and R. Govindan, "Energy-efficient rate-adaptive gps-based positioning for smartphones," in *Mobisys*, 2010.
[34] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "Unsupervised indoor localization," *Duke University*, 2012.
[35] S. Nath, "Ace: exploiting correlation for energy-efficient and continuous context sensing," in *Mobisys*, 2012.
[36] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær, "Entracked: energy-efficient robust position tracking for mobile devices," in *Mobisys*, 2009.
[37] Z. Zhuang, K. Kim, and J. Singh, "Improving energy efficiency of location sensing on smartphones," in *Mobisys*, 2010.
[38] Y. Chon, E. Talipov, H. Shin, and H. Cha, "Mobility prediction-based smartphone energy optimization for everyday location monitoring," in *SenSys*, 2011.
[39] D. H. Kim, Y. Kim, D. Estrin, and M. B. Srivastava, "Sensloc: sensing everyday places and paths using less energy," in *SenSys*, 2010.